

**REMARKS**

Reconsideration of the application as amended herein is respectfully requested. Claims 1-6 and 10 have been amended. Claims 8-9 have been cancelled without prejudice to renew in a continuation application. Claims 1-7 and 10 are pending.

**Drawings**

Attached as Exhibit A to this Response is Figure 19. Figure 19, which is referred to in the specification, was inadvertently left out of the filing of this application. Applicant respectfully submits that this drawing is supported throughout the specification, with particular reference to Paragraphs 46, 48, 49 and 50. No new matter has been added.

**Claim Rejections - 35 U.S.C. § 102(b)**

The Office Action has rejected claims 1-5 under 35 U.S.C. §102(b) as being anticipated by Koch. Applicant respectfully traverses this rejection. Koch discloses a system for breakpoint setting and detection during verification of an integrated circuit design when a behavioral simulation software program is running in the emulator. In contrast, the invention of claim 1 is directed to a system that synthesizes accessibility logic into a user's design, which increases the number of signals that can be transferred between a functional verification system and a host workstation. Applicant respectfully submits that Koch does not teach or suggest adding accessibility logic to an integrated circuit logic design. The Office Action states that the following quote from Koch teaches "synthesizing accessibility logic into the user's design":

[Source-Level Emulation] allows running hardware to be debugged symbolically, as in software debugging, including

**Amendments to the Drawings**

Attached herewith as Exhibit A is new drawing labeled as Figure 19.

Attachment (Exhibit A):      Figure 19.

examining variables, setting breakpoints, and performing single step operations. All this is possible with the application running as a real hardware implementation on a hardware emulator. By back-annotating the values read from the circuit, debugging can be done at the source-code level.

Office Action, pp. 3 (quoting from Koch at p. 210, paragraph 4, Emphasis in Office Action).

Applicant respectfully submits that neither this section nor any other in Koch teach that accessibility logic that supplements the user's logic is synthesized into the user's logic design. All this section states is that a behavioral simulator, i.e., the "application" (which is a software-based functional verification tool), is run on an emulator (a hardware-based functional verification tool). A behavioral simulator is a computer program, not logic, meaning that a behavioral simulator cannot be accessibility logic that is synthesized into the user's logic design, as is required by claim 1.

In addition to the above, Applicant respectfully submits that even if Koch discloses synthesizing accessibility logic into a user's design, which it does not, Koch does not disclose that any such accessibility logic facilitates "writing of data received from the host workstation to the memories and registers in the user's design", as is required by claim 1. Likewise, Koch does not disclose any kind of accessibility logic that facilitates "reading of data stored in the memories and registers in the user's design for transfer to the host workstation", as required by claim 1.

The Office Action states that the following text from Koch teaches that Koch contains accessibility logic that creates access ports to the memories and registers:

...and for register values read from the emulator to be related back to the source code ... All this is possible with the application running as a real hardware implementation on a hardware emulator. By back annotating the values read from the circuit, debugging can be done at the source-code level.

Office Action, pp. 4 (quoting from Koch at p. 210, paragraph 4, Emphasis in Office Action).

Neither this section nor any other in Koch say anything about adding logic to a user's design where this logic facilitates reading and writing of data. All this quote says is that register values are read and then related back to source code. A portion of the paragraph where this quote comes from that was not stated in the Office Action is also instructive because it makes clear that the "source code" referred to in Koch is the user's design (i.e., the VHDL code):

The idea is to run the application on the emulator hardware and keep track of the correlation between hardware elements and the behavioral VHDL source code, allowing the emulator's operation to be controlled at the source-code level and for ...

Koch, pp. 210. Thus, in Koch, register values read from the emulator are "related back" to the user's design (i.e., the VHDL source code). This is not the same thing as having the claimed synthesized accessibility logic reading data to and writing data from the registers and memories in the user's design for communication with a host workstation. Applicant notes that it is this accessibility logic that increases the communication bandwidth between the host computer and the logic verification system. Thus, Applicant respectfully submits that Koch does not anticipate claim 1, as it has been amended.

Claims 2-5 have been rejected as being anticipated by Koch as well. However, as seen above, claim 1, from which each of them depend, is allowable over Koch, meaning that claims 2-5 are as well.

#### **Claim Rejections - 35 U.S.C. § 103(a)**

The Office Action has rejected claims 6-7 under 35 U.S.C. § 103(a) as being unpatentable over Koch in view of Patel. Applicant respectfully traverses this rejection. As discussed above, Koch is directed to a system that discloses a system for breakpoint setting and

detection during verification of an integrated circuit design when a behavioral simulation software program is running in the emulator. In contrast, the invention of claims 6-7 relates to logic that is synthesized into a target logic circuit (i.e., the “target” of the functional verification) so that data can be efficiently written to and read from registers and memories of the target logic circuit. The various elements of claims 6-7 are directed to this and are not taught by either Koch or Patel.

First, the Office Action states that Koch discloses “command decode logic” on pages 223-225 as well as Fig. 9. The Office Action states that the “run circuit” item in Figure 9 meets this limitation. This is not correct. First, the “run circuit” item in Figure 9 of Koch is not a circuit element, but instead is one state of a state diagram. A “state” is not a tangible item that can meet the limitation of a physical element such that the “command decode logic” found in claim 6. Even if it could, the “run circuit” state merely is the state of operation where the circuit programmed into the emulator in Koch is running. Moreover, the entire section referred to in Koch, Section 5.2 relates to a debugging controller. According to Koch, the debugging controller described in that section is directed to programming breakpoints and interrupts into the circuit. In contrast, the command control logic of claim 6 decodes read and write commands located in the packets stored in the incoming packet register. Applicant respectfully submits that these read and write commands are not breakpoints and interrupts, but are instructions to read data from or write data to the registers and memories in the target logic system. Moreover, Koch says nothing about “packets” containing data and commands, meaning that Koch cannot teach or suggest “command control logic” like that of claim 6.

The Office Action further argues that Koch discloses “write command execution logic” in the same sections discussed above. As discussed, Figure 9 in Koch is a state diagram, not a

circuit, meaning that it does not describe anything tangible. Even if such a state diagram teaches elements of a circuit, Applicant respectfully submits that the “write to breakpoint” state in Figure 9 of Koch does not teach the claimed “write command execution logic” of claim 6. Claim 6 makes it clear that the “write command execution logic” executes the write commands decoded from the packets sent to the packet register from the host workstation so that data in the packets is written into the appropriate register or memory location. In contrast, the “write to breakpoint” state described in Koch relates to a command executed by the debug controller. The command is not part of a packet of data sent from the workstation and stored in a packet register..

The Office Action also argues that Koch teaches “read command execution logic” in Section 5.2 of Koch. Applicant respectfully disagrees. As with the other elements of the claim, the Office Action refers to the state diagram on page 224 and argues that the “read data path registers” state teaches the “read command execution logic” limitation of claim 6. As discussed above, this is not accurate, as the “state” shown in the state diagram is not a tangible circuit, as required by claim 6. Regardless, Koch does not teach that the “read data path registers” state disclosed therein acts to store the information read registers in an “outgoing packet register”, as required by claim 6. In fact, neither Koch nor any of the other cited references say anything about a structure such as an “outgoing packet register.

Finally, the Office Action states that Koch discloses “interface logic” on pages 223-224 and in Figs. 9 and 10a. Specifically, the Office Action states that in Koch, “[e]ach register in the data path is replaced as shown in Figure 10 by a register that allows the required control of the operation.”. Applicant respectfully submits that replacing registers in the data paths with a different type of registers is not “interface logic”. First, in claim 6, the “registers” and “interface logic” are separate claim elements. Claim 6 says nothing about replacing the registers in the

target logic circuit design with different registers, meaning that claim 6 requires both “register” and “interface logic” structures. Under the Office Action’s construction, the registers in the claim and the interface logic would have to be the same thing, which is not a proper construction.

The Office Action acknowledges that Koch does not teach or suggest “protocol logic” and the incoming and outgoing packet registers that form part of this protocol logic. The Office Action states that these features are found in Figure 9A, as well as at Col. 23, line 63, through Col. 24, line 14. The Office Action states:

Patel, on the other hand, does expressly teach the use of “Hardware Buffers and Latches” that store incoming and outgoing data signals in the communication between a “Reference Element” (which corresponds to the claimed “protocol logic”) and an “Arbitration and Synchronization Circuit” (which corresponds to the claimed “host workstation”).

Applicant respectfully submits that this argument is not correct. Firstly, claim 6 makes it clear that the packet registers store “packets” and that the packets include data from the host computer as well as commands. According to claim 6, these commands include read and write commands. Patel says nothing about packets and thus nothing about these packets containing data and commands.

In order for a combination of reference to render a claim unpatentable, the combination must teach each and every claim limitation. As seen above, neither Koch nor Patel supply all of the limitations present in claim 6. In fact, as seen above, Koch and Patel teach almost nothing claimed in claim 6. Thus, Applicant respectfully submits that claim 6 is allowable over the combination of Koch and Patel.

Because claim 6 is patentable over Koch in view of Patel, Applicant respectfully submits that claim 7, which is dependent upon claim 6, is allowable as well. No further comment is deemed necessary.

The Office Action has reject claim 8 under 35 U.S.C. § 103(a) as being unpatentable over Koch in view of X.25. Applicant has cancelled claim 8 without prejudice to renew in a continuation application.

The Office Action has rejected claim 9 under 35 U.S.C. § 103(a) as being unpatentable over Koch in view of X.25 and further in view of Patel. Applicant has cancelled claim 9 without prejudice to renew in a continuation application. No further comment is deemed necessary.

### **Claim Objections**

The Office Action has objected to claim 10 as being dependent upon a rejected base claim. The Office Action states that rewriting claim 10 including all the limitations from the base claims would place claim 10 in condition for allowance. Applicant has so amended claim 10, thereby placing claim 10 in condition for allowance.

### **CONCLUSION**


In view of the foregoing, Applicant respectfully submits that the present application is in condition for allowance, which is respectfully requested. If the Examiner believes that a telephone conference would be useful in moving the application forward to allowance, the Examiner is encouraged to contact the undersigned at (650) 614-7400. If additional fees are needed, the Office is authorized to charge Deposit Account No. 15-0665.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: February 18, 2005

By:

A handwritten signature in black ink, appearing to read 'Jeffrey A. Miller', is written over a horizontal line.

Jeffrey A. Miller  
Reg. No. 35,287

Four Park Plaza, Suite 1600  
Irvine, California 92614-2558  
(650) 614-7660